Seminario de Investigación 17-18

Fireworks Algorithm applied to Smart Cities field & Optimizations

Universidad Politécnica de Madrid

ETSI SISTEMAS INFORMÁTICOS

PhD Candidate: **Clemencio Morales Lucas**

PhD Director: **Dr. Luis Fernando de Mingo López**

Index of Contents

Index of Contents

Q&A (anytime)

Index of Contents

1. The Natural Computing Paradigm

1. The Natural Computing Paradigm (II)

- Natural Computing encloses three classes of algorithms:

1

- Those that take inspiration from nature for the development of novel problem-solving techniques.

2

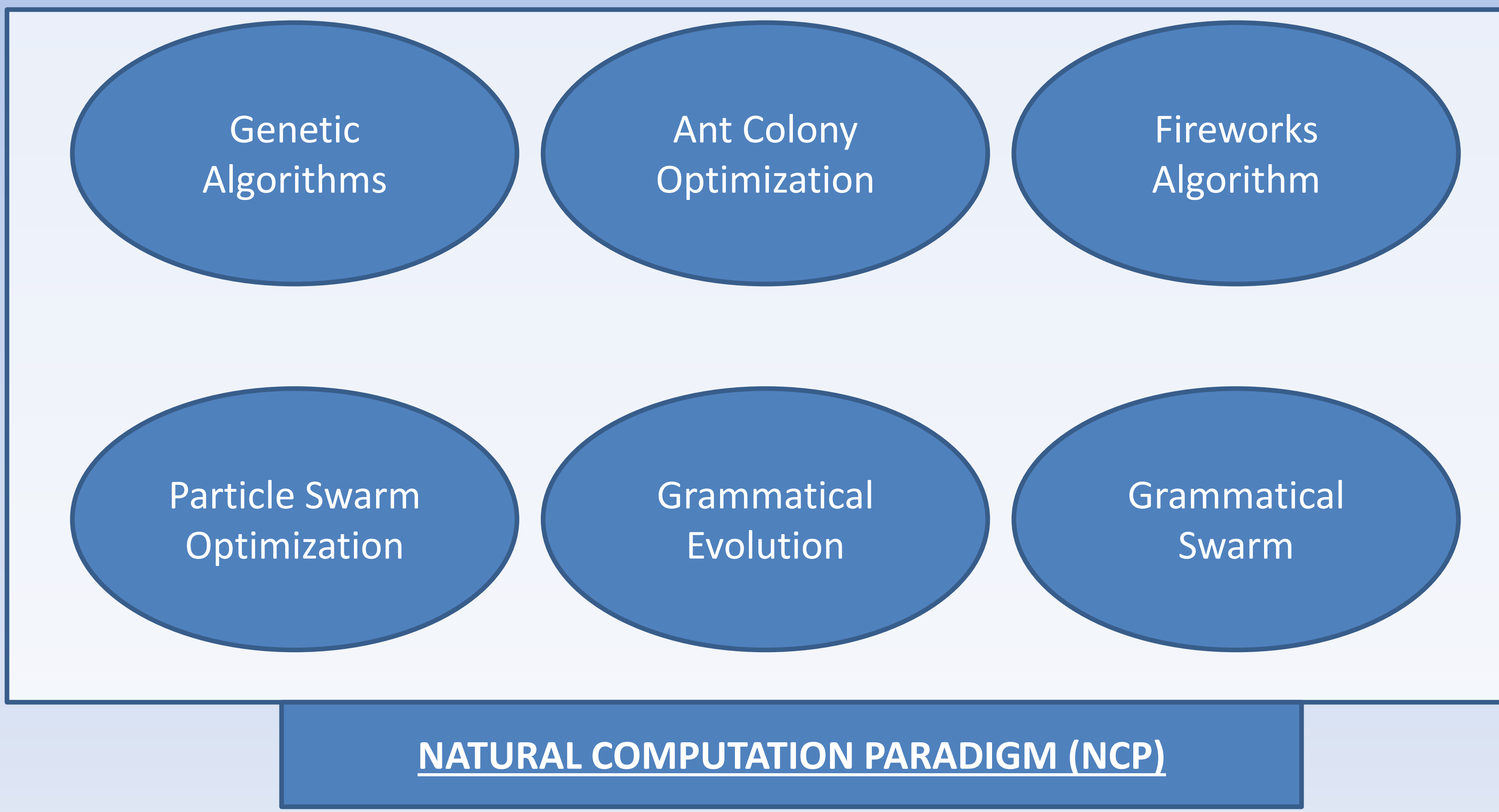
- Those that are based on the use of computers to synthesize natural phenomena.

3

- Those that employ natural materials, such DNA or molecules, to compute.

1. The Natural Computing Paradigm (III)

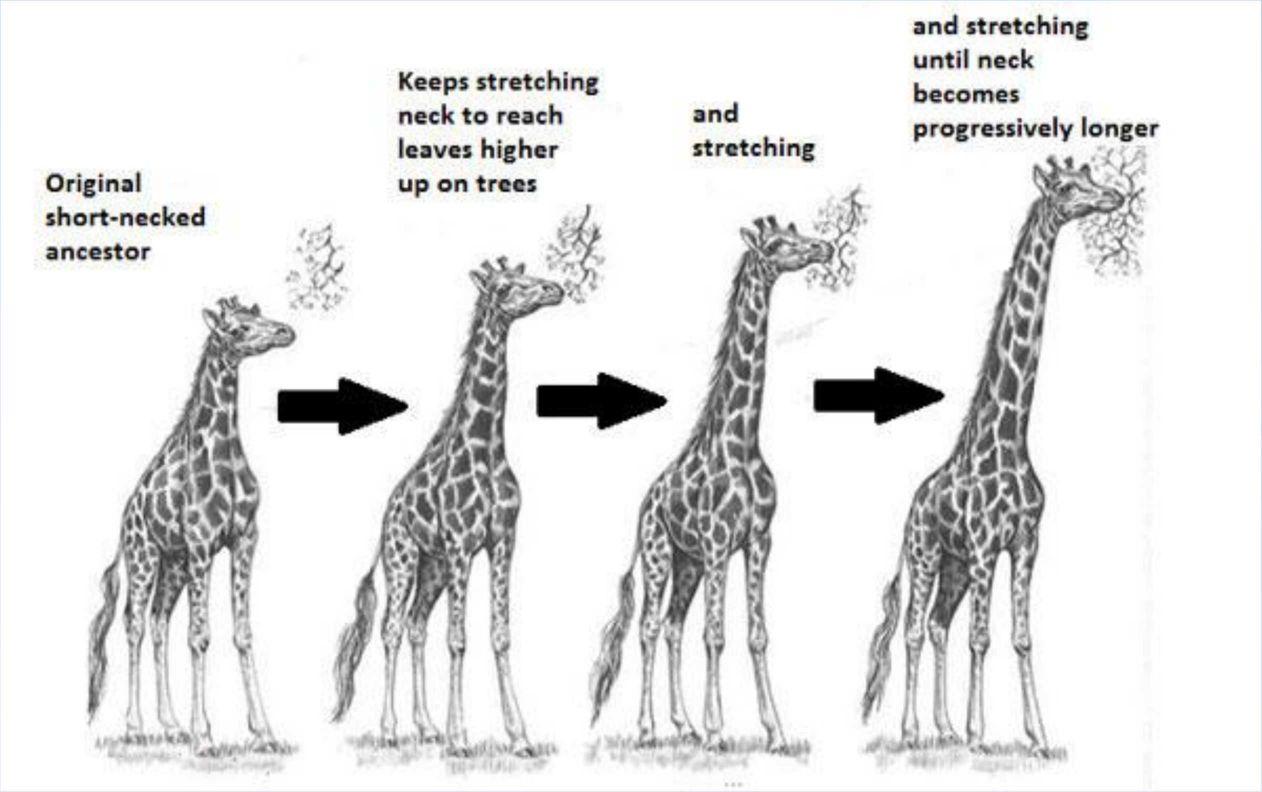
- Natural Computing spectrum:



1. The Natural Computing Paradigm (IV)

- Darwin, 1859: “*Living beings have been **forced** to a continuous **evolutive process** looking for **survival***”

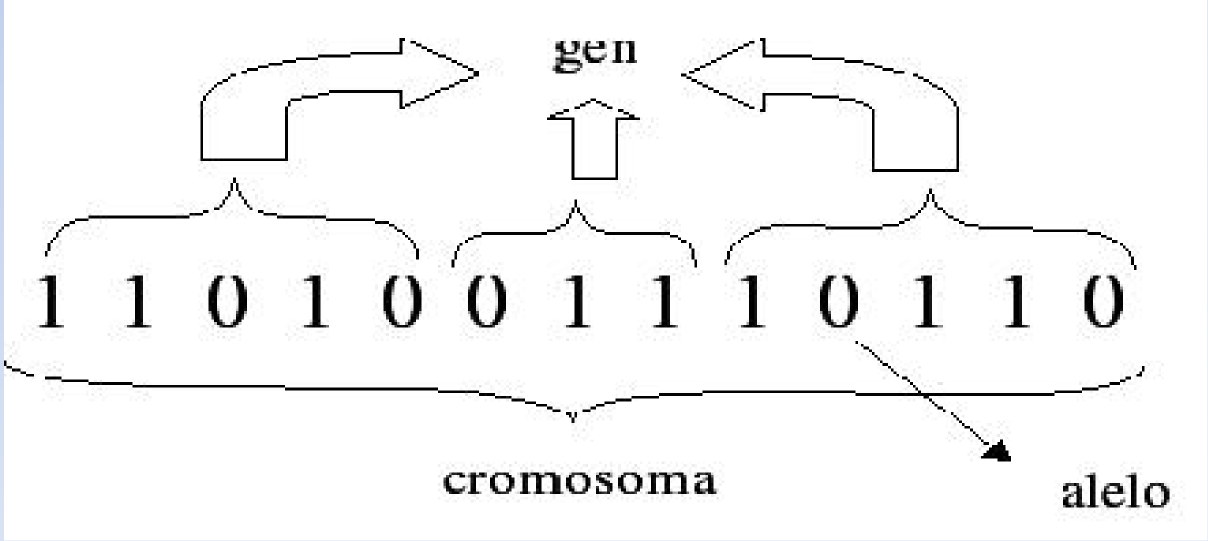
- Natural Selection & Evolution:



1. The Natural Computing Paradigm (V)

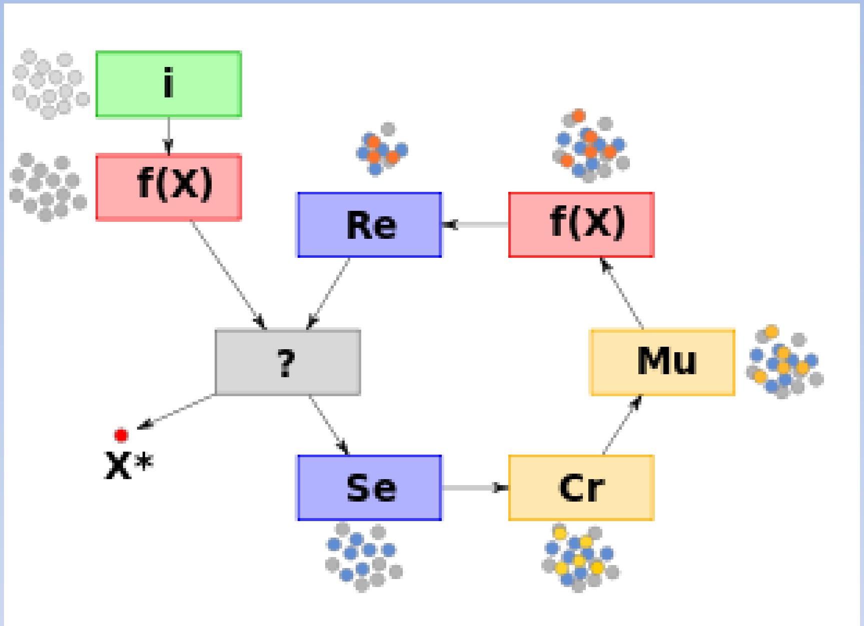
- GA: *Set of ordered instructions, that aim to **evolve** a **population** to reach a **solution** (improved population)*

- Genetic Algorithms (GA) have the following set of atomic elements:



1. The Natural Computing Paradigm (VI)

- Genetic Algorithms flux diagram:

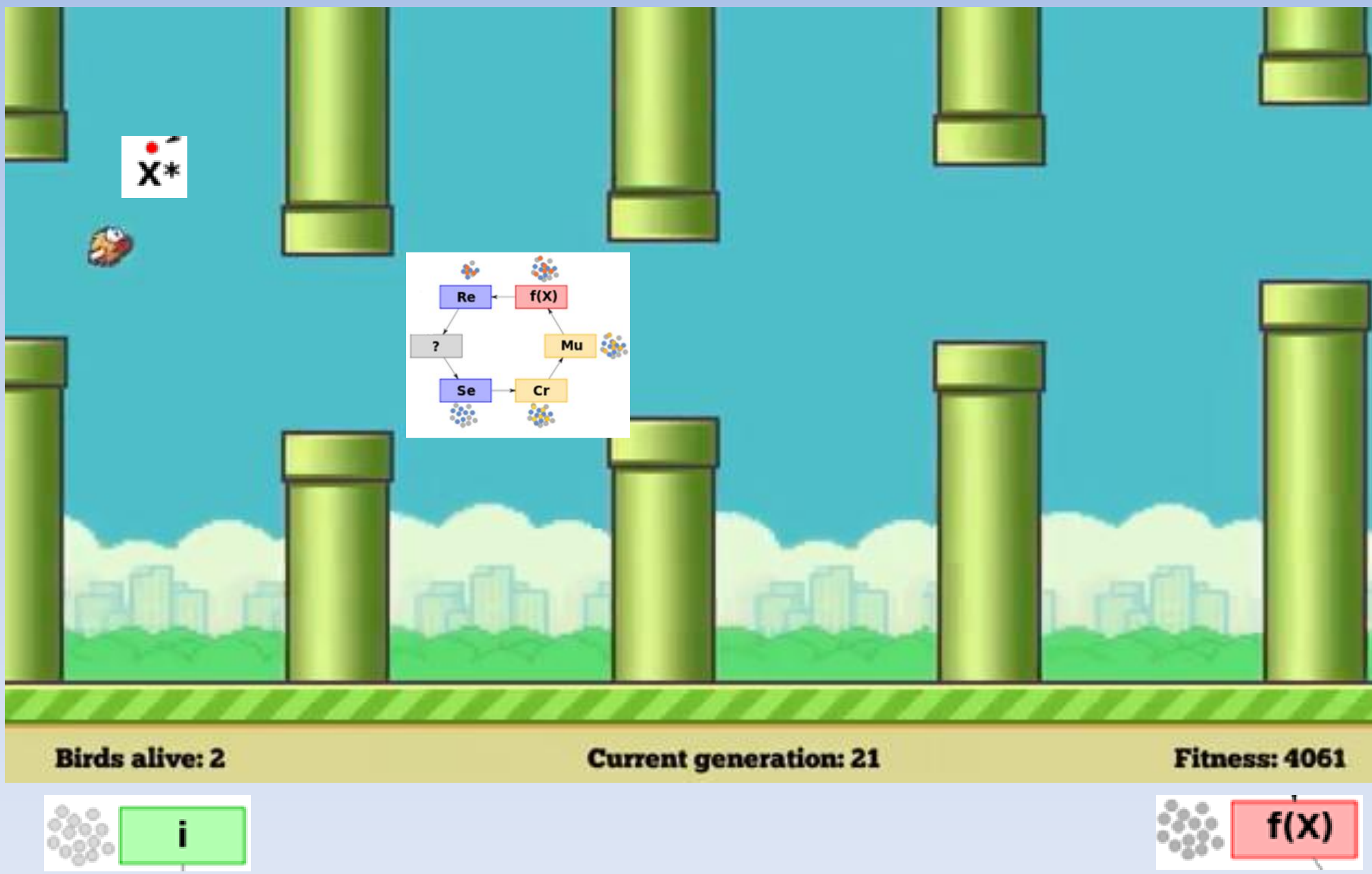


- Selection: *Truncation* approach recommended:

$$\Omega_T^*(s,T)(f_i) = s^*(f_i) = \begin{cases} 0 & if\ S(f_i) \le (1-T)N \\ \dfrac{S(f_i)-(1-T)N}{T} & if\ S(f_i-1) \le (1-T)N < S(f_i) \\ \dfrac{s(f_i)}{T} & else \end{cases}$$

1. The Natural Computing Paradigm (VII)

- Genetic Algorithm sections for Flappy Bird Algorithm:



1. The Natural Computing Paradigm (VIII)

- PSO: Reynolds, 1987: "*Technique to optimize a problem due to a **meta-heuristic** strategy. Iterative improvement of a candidate solution with regards to a pre-stipulated quality criteria*"

1. The Natural Computing Paradigm (IX)

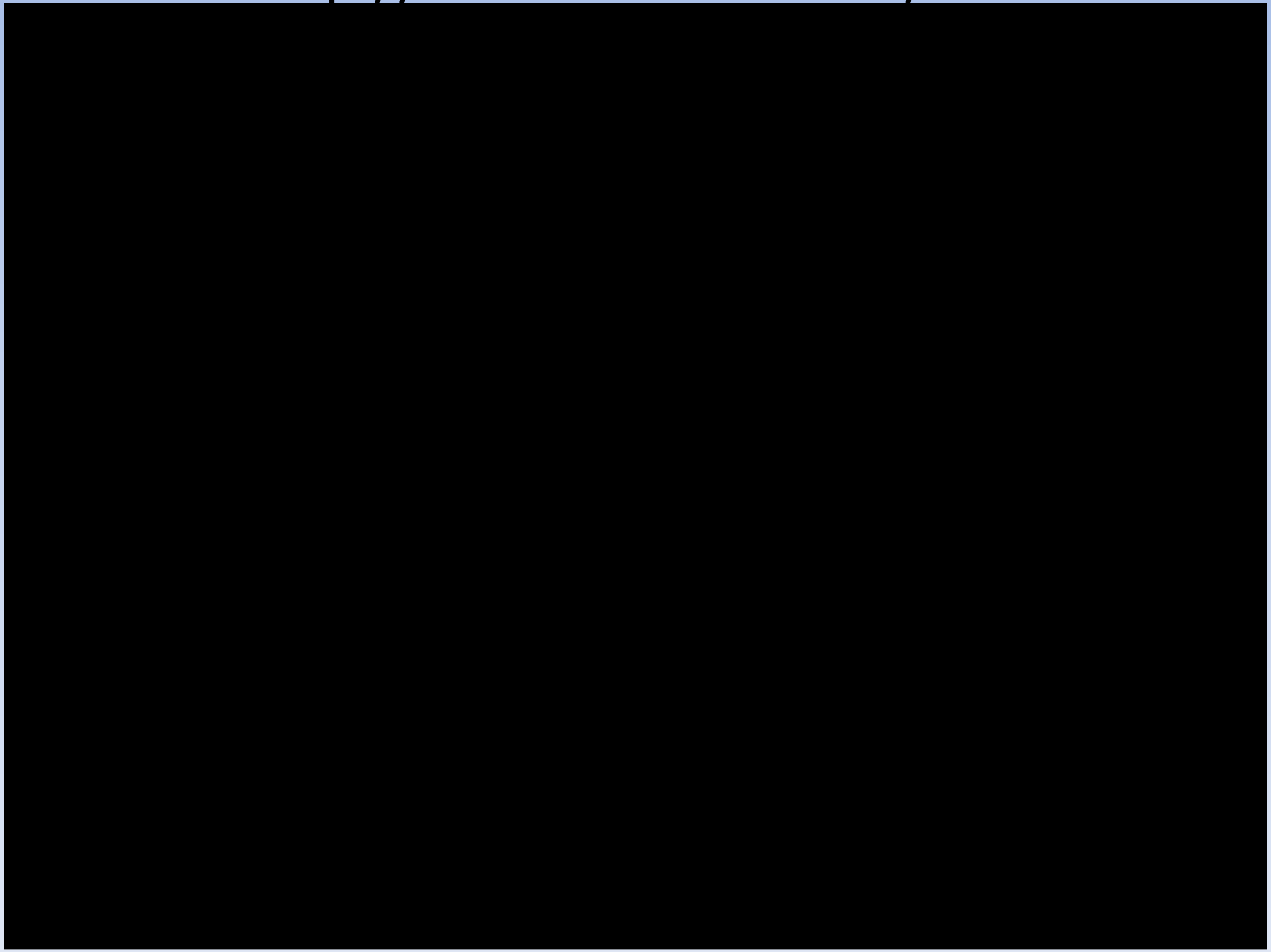
- Particle Swarm Optimization (PSO) pseudo-algorithm:

```
for each (particle within S){
  position = generateRandomValue(S[i], b_low, b_up);
  position = bestKnownPositionByParticle(S[i]);
  if( f(p) < f(g) ){
    bestGlobalPosition = position;
  }
  speed = generateRandomSpeed(b_low - b_up, b_low - b_up)
 }

while(!stopCriteria){
  for each (particle within S){
    for each (dimension within d){
      first_op = ωv_(i,d)+ φ_p * r_p;
      second_ op = (p_(i,d)- x_(i,d);
      third_ op = φ_g * r_g * (g_d- x_(i,d);
      d[i] = first_op * second_op + third_op;
    }
    Position += d[i];
    if( f(xi) < f(pi) ){
      bestParticleLocalPosition = xi;
      if( f(pi) < f(g)){
        bestGlobalPosition = pi;
      }
    }
  }
return bestGlobalPosition;
}
```

1. The Natural Computing Paradigm (X)

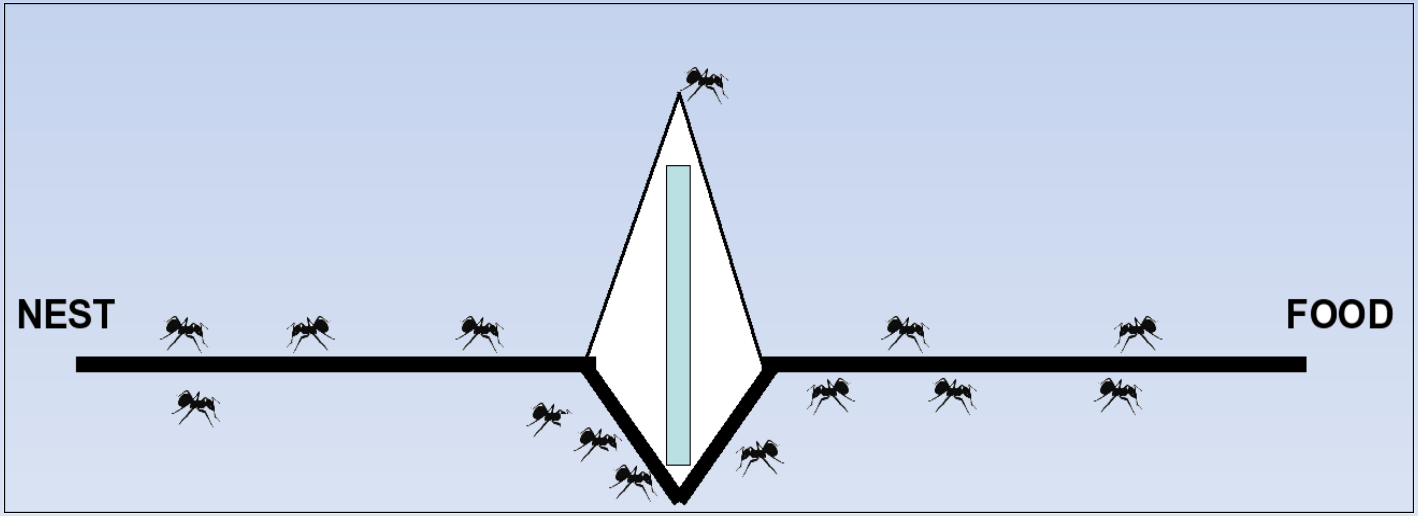
http://www.realflow.com/



1. The Natural Computing Paradigm (XI)

- Dorigo & Di Caro, 1992: “*Multi-agent paradigm inspired on the ants idiosyncrasy when searching for livelihood*”

- Goss Experiment with *Iridomyrmex humilis* colony:



1. The Natural Computing Paradigm (XII)

- How do all the ants ***know*** what is the shortest path?

- **Stigmergy**: Collaboration protocol where communication is made due to the accumulation of objects, such pheromones

$$p_{ij}^{k} \begin{cases} \tau_{ij} \; si \; j \in N_i \\ \\ 0 \; si \; j \notin N_i \end{cases}$$

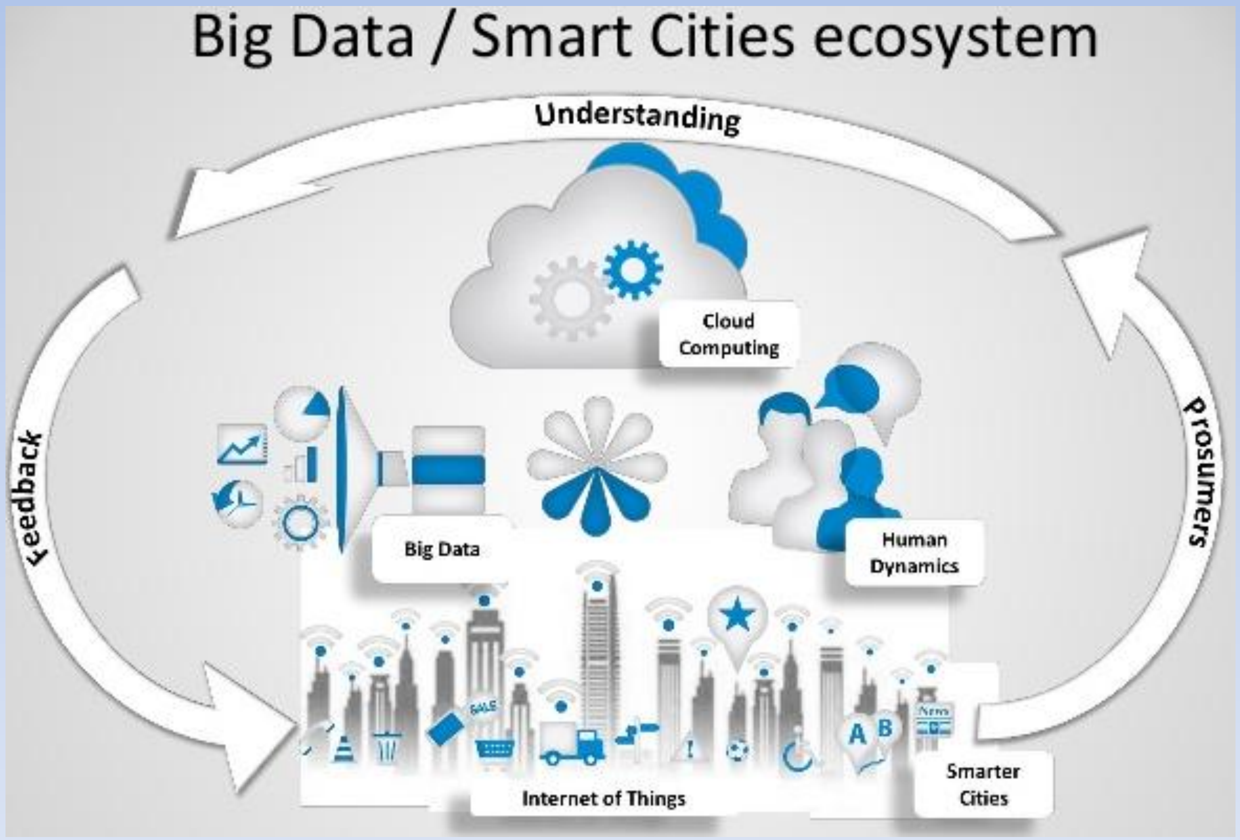
1. The Natural Computing Paradigm (XIII)

- Urban cores massification→ Urgent **improvement** needed

Smart Public Services ✓

Smart Mobility ✓

Smart Water



Smart Buildings and Homes ✓

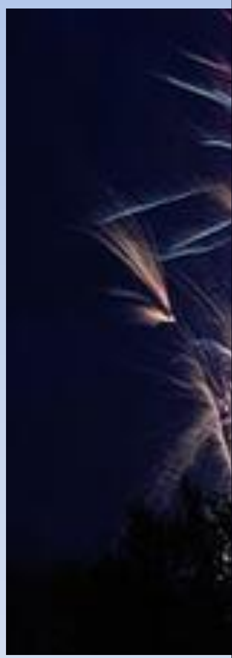
Smart Integration ✓

Smart Energy

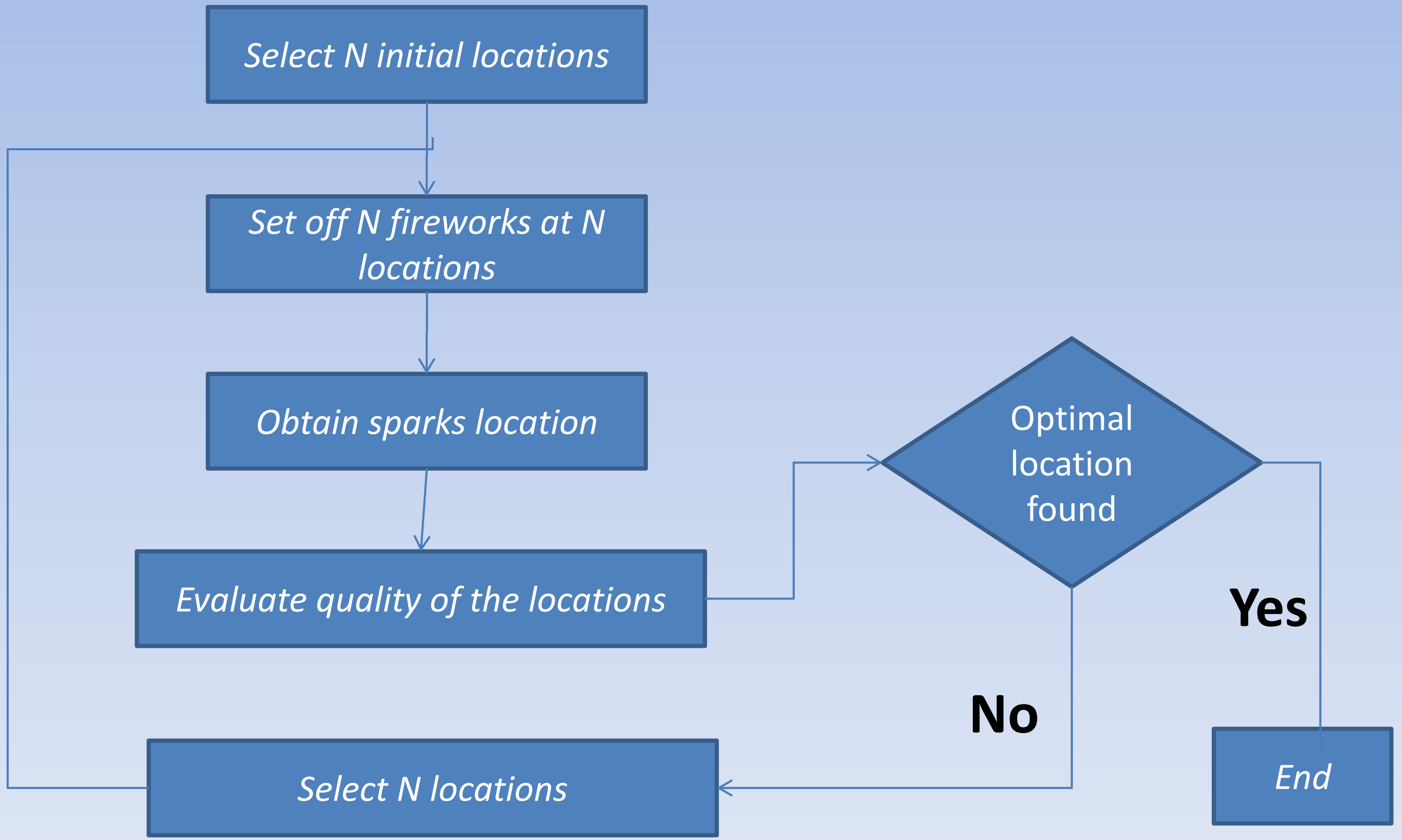
Index of Contents

2. Fireworks Algorithm

- FWA: Incremental and iterative search process in a huge solution space.

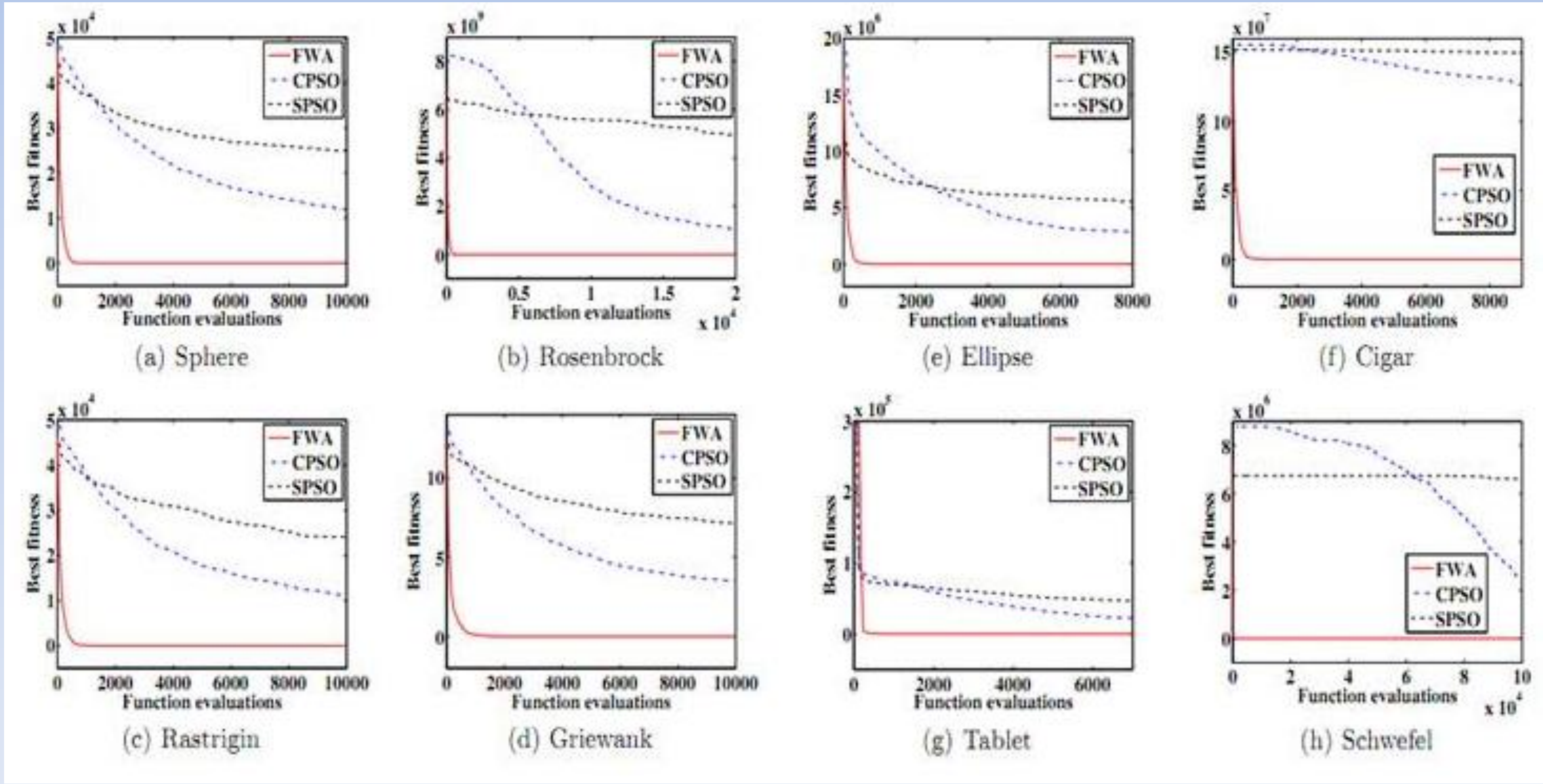


2. Fireworks Algorithm (II)



2. Fireworks Algorithm (III)

- **FWA** **versus** **CPSO** (*Combinatorial Particle Swarm Optimization*) & **SPSO** (*Standard Particle Swarm Optimization*) → 8 Benchmark *f(x)* over 20 runs:



2. Fireworks Algorithm (IV)

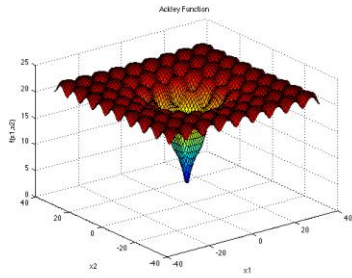
- $\bar{x}$ and $\sigma_{\bar{x}}$ for **FWA**, **CPSO** and **SPSO** on 9 benchmark *f(x)*→20 independent runs of 10000 *f(x)* evaluations:

Function	FA's mean (StD)	CPSO's mean (StD)	SPSO's mean (StD)
Sphere	0.000000 (0.000000)	11857.425781 (3305.973067)	24919.099609 (3383.241523)
Rosenbrock	19.38330 (11.94373)	2750997504.000000 (1741747548.420642)	5571942400.000000 (960421617.568024)
Rastrigrin	0.000000 (0.000000)	10940.148438 (3663.484331)	24013.001953 (4246.961530)
Griewank	0.000000 (0.000000)	3.457273 (0.911027)	7.125976 (0.965788)
Ellipse	0.000000 (0.000000)	2493945.500000 (1199024.648305)	5305106.500000 (1117954.409340)
Cigar	0.000000 (0.000000)	122527168.000000 (28596381.089661)	149600864.000000 (13093322.778560)
Tablet	0.000000 (0.000000)	15595.107422 (8086.792234)	42547.488281 (8232.221882)
Schwefel	4.353733 (1.479332)	8775860.000000 (1217609.288290)	6743699.000000 (597770.084232)
Ackley	0.000000 (0.000000)	15.907665 (1.196082)	18.423347 (0.503372)

Index of Contents

3. Current Research

- Currently centered on Fireworks Algorithm
- Eleven Benchmarking functions for testing algorithm's performance




```java
private double ackleyFunction(final double[] inputValues) {
    //Ackley Function Bounds[-32,32]   dim30 optimum=0D ini[16,32]
    double aux = 0, result = 0;
    for (int i = 0; i < inputValues.length; i++) {
        result += Math.pow(inputValues[i], DOUBLE_SQUARE_VALUE);
        aux += Math.cos(SQUARE_VALUE * Math.PI * inputValues[i]);
    }
    result = ACKLEY_FUNCTION_NEGATIVE_MULTIPLIER * Math.sqrt(result / inputValues.length);
    result = ACKLEY_FUNCTION_LOWER_BOUND * Math.exp(result);

    aux = -Math.exp(aux / inputValues.length);

    result += ACKLEY_FUNCTION_UPPER_BOUND + Math.exp(DOUBLE_IDENTITY_VALUE) + aux;
    return result;
}
```

$$f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

3. Current Research (II)

- Pre-initialization for Spark generation in FWA
- Hardware warm-up phase for accurate results
- Executing FWA to optimize mathematical functions
- Gaussian Explode (adding a random value from a Gaussian Distribution to each spark’s direction a new area of interest) for FWA Sparks

3. Current Research (III)

- Classic FWA & Pre-initialized FWA are launched

```java
private void launchFWAForAllParameters() {
    for (int i = 1; i <= BenchmarkFunctionConstants.NUMBER_OF_FUNCTIONS; i++) {
        System.out.print(FITNESS + i + SEPARATOR);
        maximumBound = new double[availableDimensions[i - 1]];
        minimumBound = new double[availableDimensions[i - 1]];
        for (int j = 0; j < maximumBound.length; j++) {
            maximumBound[j] = availableBounds[i - 1];
            minimumBound[j] = -availableBounds[i - 1];
        }
        for (int k = 0; k < shiftIndex.length; k++) {
            BenchmarkFunction benchmarkFunction = new BenchmarkFunction();
            benchmarkFunction.setIndexAndShift(i, availableBounds[i - 1] * shiftIndex[k]);
            double avg = 0;
            for (int t = 0; t < NUMBER_OF_ITERATIONS; t++) {
                FireworkAlgorithm fireworkAlgorithm = new FireworkAlgorithm(LOCATIONS_NUMBER, NUMBER_OF_SPARKS,
                        LOW_BOUND_NUMBER, HIGH_BOUND_NUMBER, MAXIMUM_AMPLITUDE_VALUE, GAUSSIAN_SPARKS_VALUE,
                        maximumBound, minimumBound, filePath, benchmarkFunction);
                avg += fireworkAlgorithm.launch();
            }
            avg /= NUMBER_OF_ITERATIONS;
            System.out.print("\t" + avg);
        }
        System.out.println();
    }
}
```

Index of Contents

4. Investigation Results

- Classic FWA for 6 Shifts (iterations) in 12 Fitness calculations:

```
"C:\Program Files\Java\jdk1.8.0_91\bin\java" ...
Average values: |Shift 0|   |Shift 1|   |Shift 2|   |Shift 3|   |Shift 4|   |Shift 5|   |Shift 6|
|Fitness 1|     87964.53164539687   115177.79823411905  104047.94466293606  152832.21793368732  181590.9425124937   186796.91465742205  338939.0914984318
|Fitness 2|     1304626.5037893485  1183071.1769431357  1569887.8714619293  2248142.425844039   2156187.912805617   2551900.305705179   4877537.981555655
|Fitness 3|     2.269935022878593E8 3.8374710420193833E8    6.323412185512973E8 1.0303994908085098E9    1.3248723550103106E9    1.7579998766181521E9    4.770778039461274E9
|Fitness 4|     20.40234322151514   20.9222973702228    20.971153395533328  21.04318844908315   21.21092141469584   21.27371661806913   21.417119430726906
|Fitness 5|     641.0013762555989   893.5523453874546   1135.7937523181397  1325.7595146264614  1571.006993633494   1613.5773149214879  2931.0896486873544
|Fitness 6|     381.3537961159256   427.50613436117953  429.4147060218793   470.1477167072838 5  566.5988604748532   646.216163593115    766.0707186496988
|Fitness 7|     1.6135901299819527E9    2.756045176695660 6E9    3.619054718223214E9 5.300643087917137E9 8.291791801678309E9 8.902541812263472E9 3.214332690322102E10
|Fitness 8|     268.839156631857    529.6218899649532   897.5575816321068   1944.987147494997   4222.162790914868   12495.367349302887  31151.973415354372
|Fitness 9|     456.5756115310787   466.63102033104667  570.9086964049048   824.8189967236536   4271.746761233097   5251.288856481333   13917.672801289393
|Fitness 10|       0.48998373042214693 0.49408117056342105 0.49600369580360243 0.4971115478474788  0.4986556274517272  0.49945212111296033 0.4997352055219162
|Fitness 11|       1094.9034304350432  1828.51066233214    1856.1030615907064  2589.466402662284   2921.865267101023   5325.282142480932   6674.375264958745
|Fitness 12|       4364143.033557112   3262484.249666566   5885663.6474779565  8102783.023780778   9672840.769236265   1.6886292769517966E7    3.3040112192410313E7
====================================================================================================================
Time elapsed for current algorithm (ns): 7572587531
====================================================================================================================
```

4. Investigation Results (II)

- Pre-Initialized FWA for 6 Shifts (iterations) in 12 Fitness calculations:

Average values:	Shift 0	Shift 1	Shift 2	Shift 3	Shift 4	Shift 5	Shift 6
Fitness 1	78630.9791875285	104210.02865264642	119887.88049383581	131573.81310339886	172176.66069728768	197438.2713592046	337108.7771787091
Fitness 2	1378036.518605469	1217938.5060555486	1515821.1063623482	2043632.3654345162	2619227.542472978	2532199.6957563036	4618281.280612802
Fitness 3	2.5561141462213868E8	4.0098386511206186E8	6.735625577396677E8	8.652053362234106E8	1.2071042563487446E9	1.6962818423289523E9	4.799203363670427E9
Fitness 4	20.730119395915636	20.524953910342013	20.971268585358523	20.89547017897334	21.21645853130096	21.32558166205632	21.396653680555897
Fitness 5	792.1730430304727	840.2983710982547	832.827144019712	1438.0666596475276	1602.738331579155	1573.875289039541	2749.588291524064
Fitness 6	358.9404765453052	400.706842766287	422.37866903685915	445.6142116617796	518.404233213165	621.296240306615	811.9978132870077
Fitness 7	1.449632910076408E9	2.4884635165290046E9	3.0451242381934795E9	6.103656020653241E9	7.962119953095451E9	9.430240181799067E9	3.2510918379741955E10
Fitness 8	186.15021505013607	389.8307687869191	827.4399310808482	2085.056162119717	4147.594493531137	12478.757307085118	30063.876071513805
Fitness 9	435.11230316609016	395.9472983641667	694.5425104795041	1312.6807786708835	2382.459722232938	6994.744443173751	14924.330772122154
Fitness 10	0.4916679089576662	0.4941638551640885	0.49552140619754487	0.49719329314183547	0.4985135942171481	0.49950486984520515	0.4997152916199254
Fitness 11	1296.748181125542	2142.1824106979825	2048.0061885018386	2943.4189306582275	3284.7266764467795	5165.909916610583	8275.061493082163
Fitness 12	3896381.226555627	5778834.4065759005	7990955.610401971	8501291.784193296	9187848.875311902	1.5255656354759997E7	4.1041761097279236E7

Time elapsed for improved algorithm (ns): 6095082401

4. Investigation Results (III)

- Ahmdal’s Law →

$$\text{Improvement (\%)} = \frac{\text{Non-Improved Time}}{\text{Improved Time}}$$

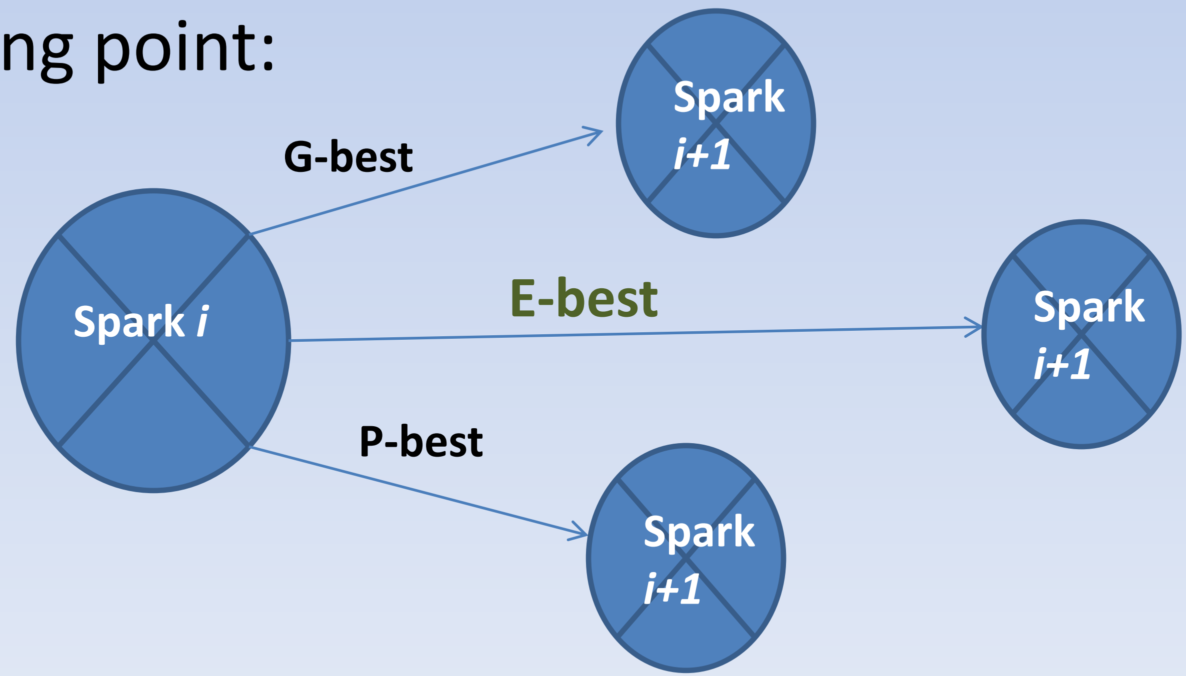
```
===============================================================================================
Time elapsed for improved algorithm (ns): 6095082401
====================================================RESULTS====================================
Ahmdal's Law = Non-Improved time /  Improved Time = 7572587531/6095082401 = 124.24093773953886
The improved algorithm is a 24.240937739538865% faster
```

4. Investigation Results (IV)

- Next investigation path: Searching for a synergy between CPSO & FWA.
- In CPSO, Global Best (*g-best*) and Personal Best (*p-best*) are taken into computation.

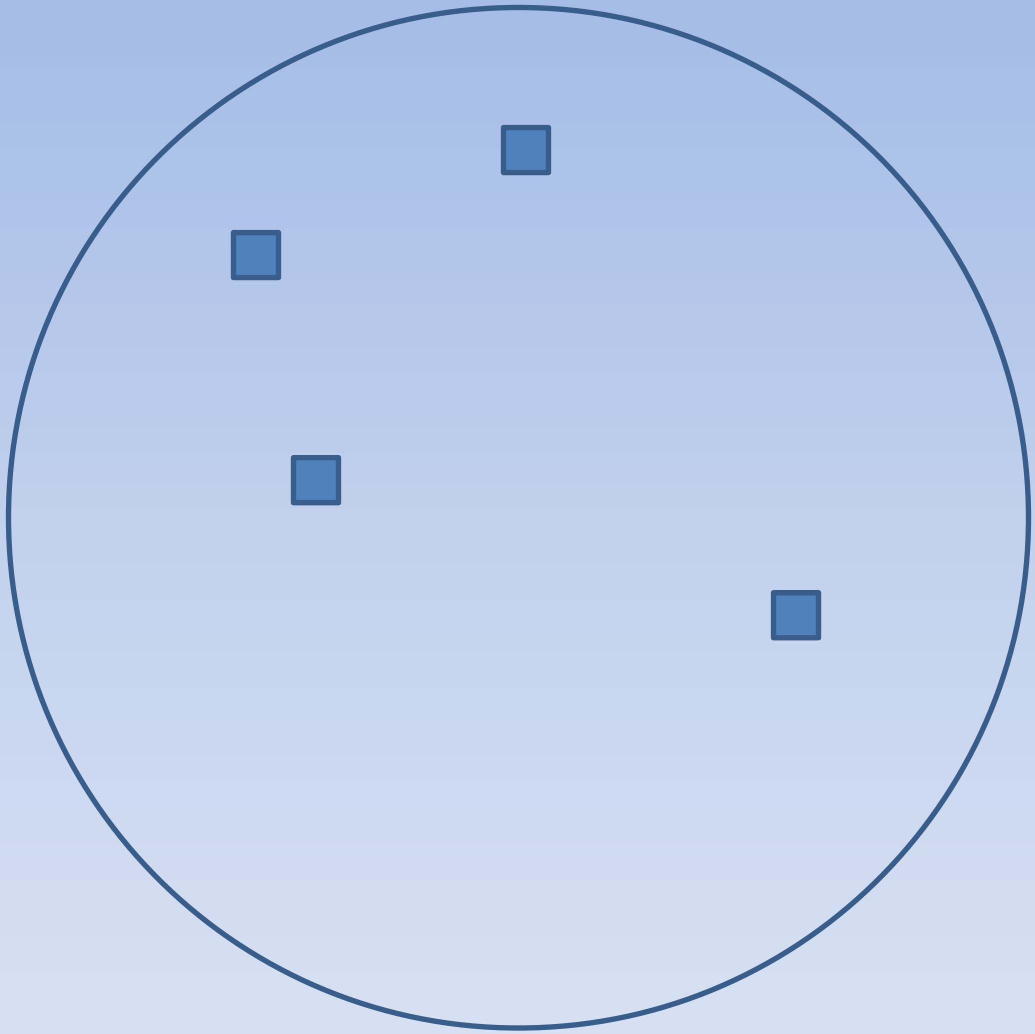
4. Investigation Results (V)

- Global Best (*g-best*) and Personal Best (*p-best*) just take swarm-related data.
- In Nature, the environment plays a central role → **Extrapolation to CPSO & FWA adding E-best**:
- A spark explosion success determines next spark's exploding point:

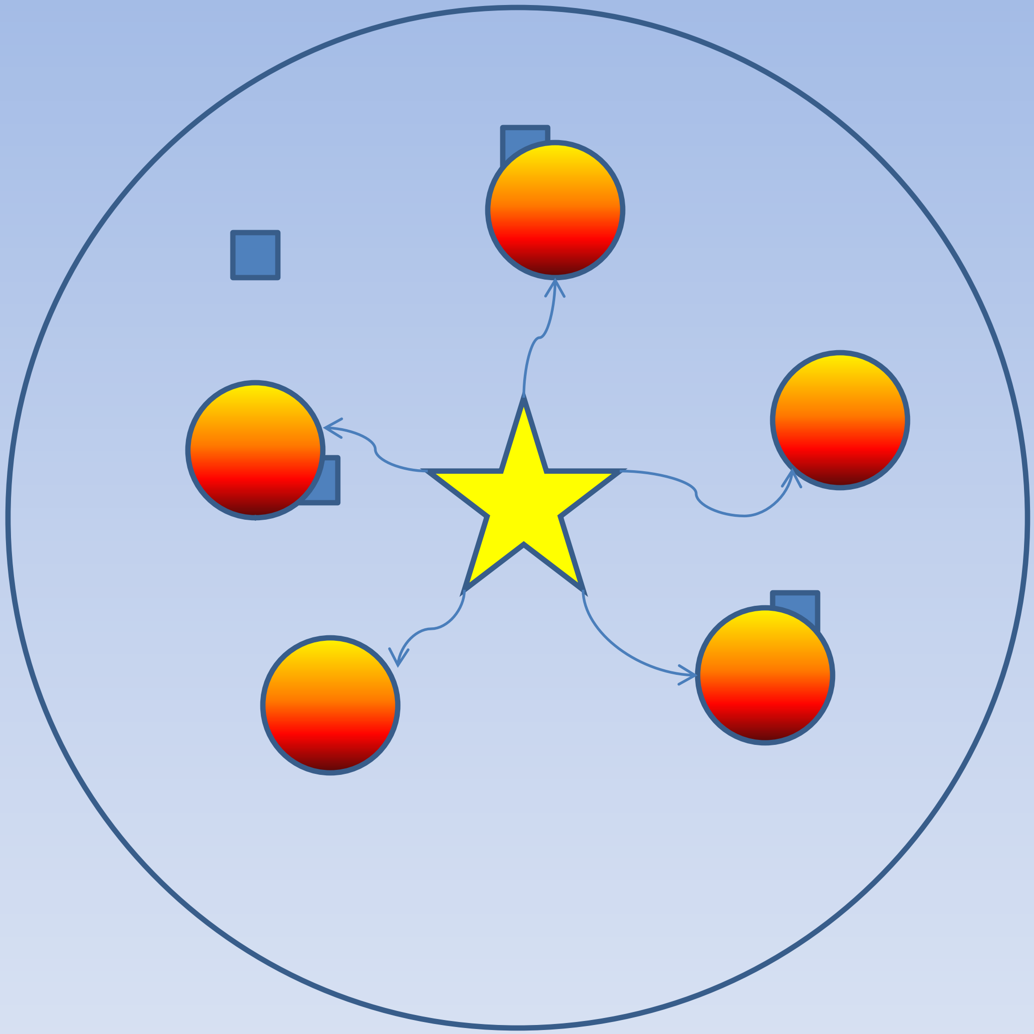


Index of Contents

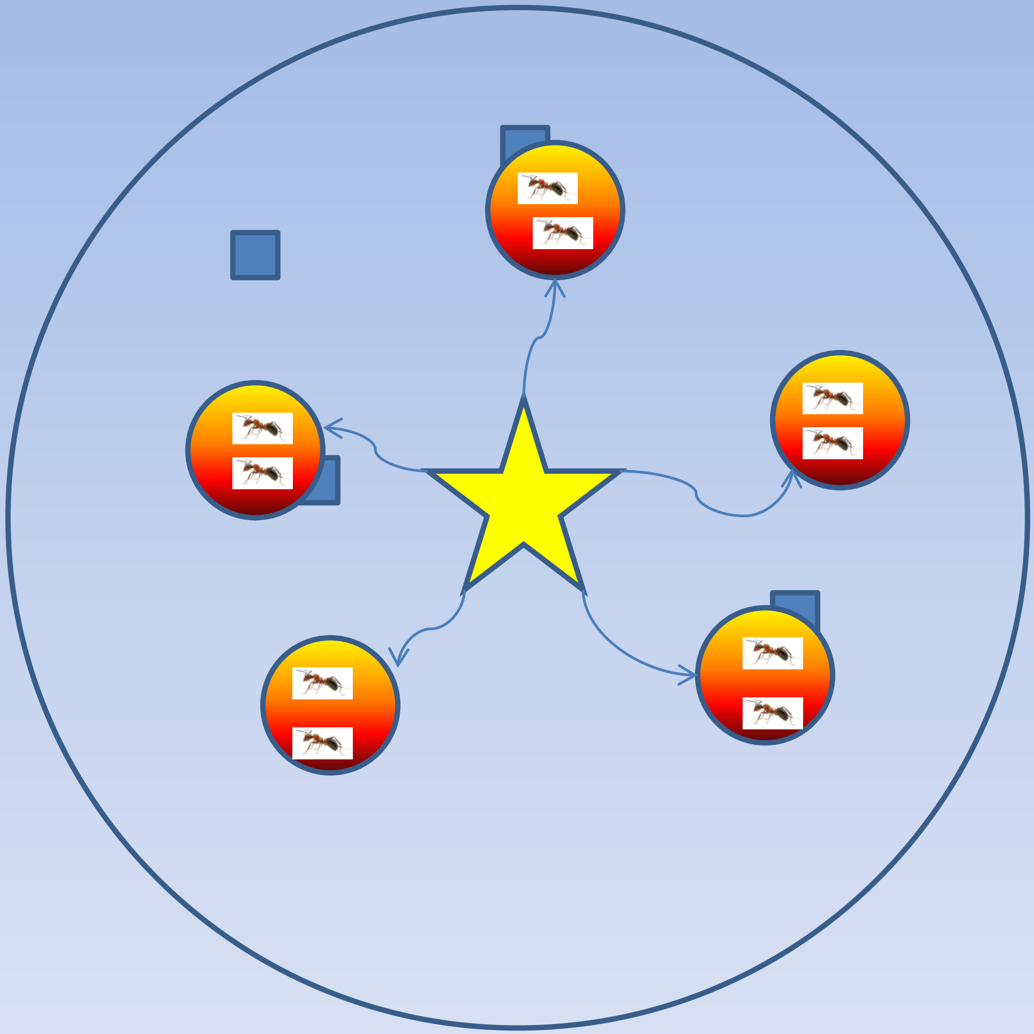
FWA+ACO



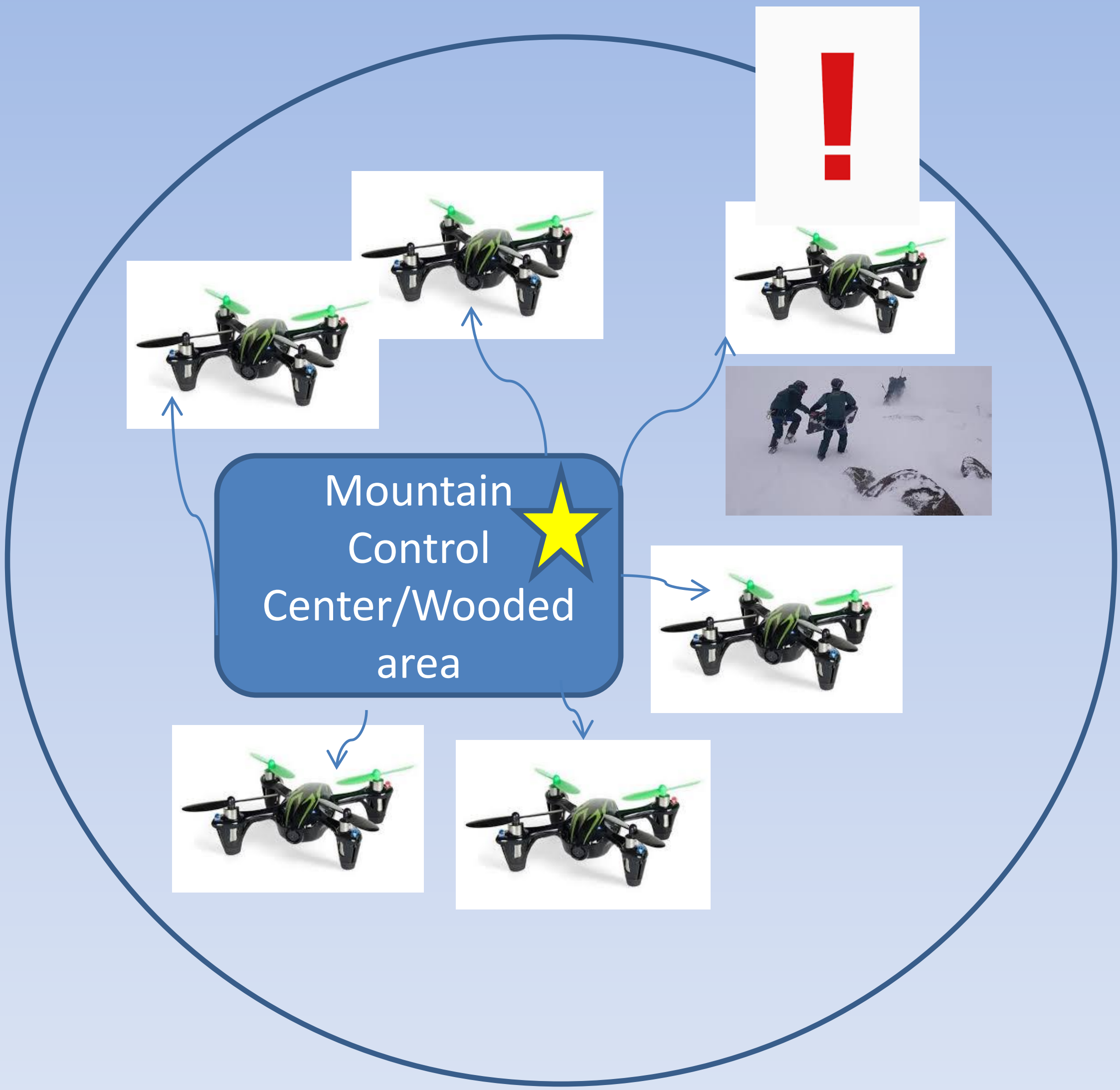
FWA+ACO (II)



FWA+ACO (III)

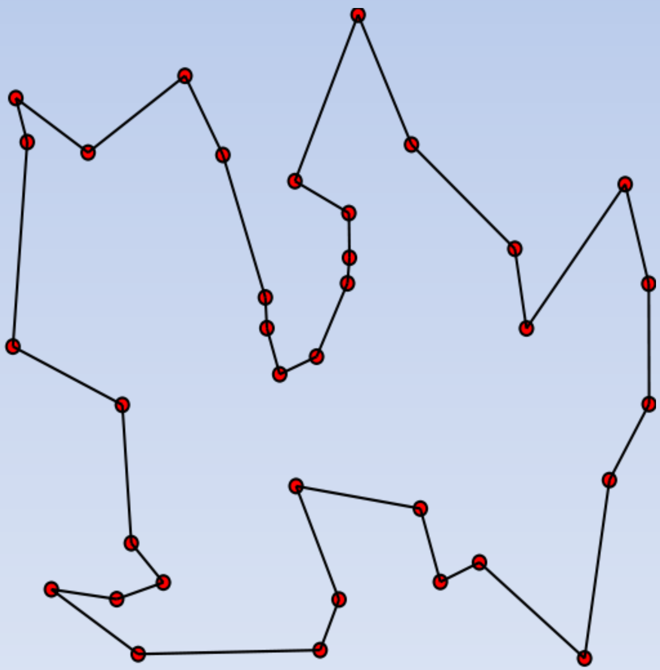


5. Feasible application for Smart City rural area



5. Feasible application for Smart City (II)

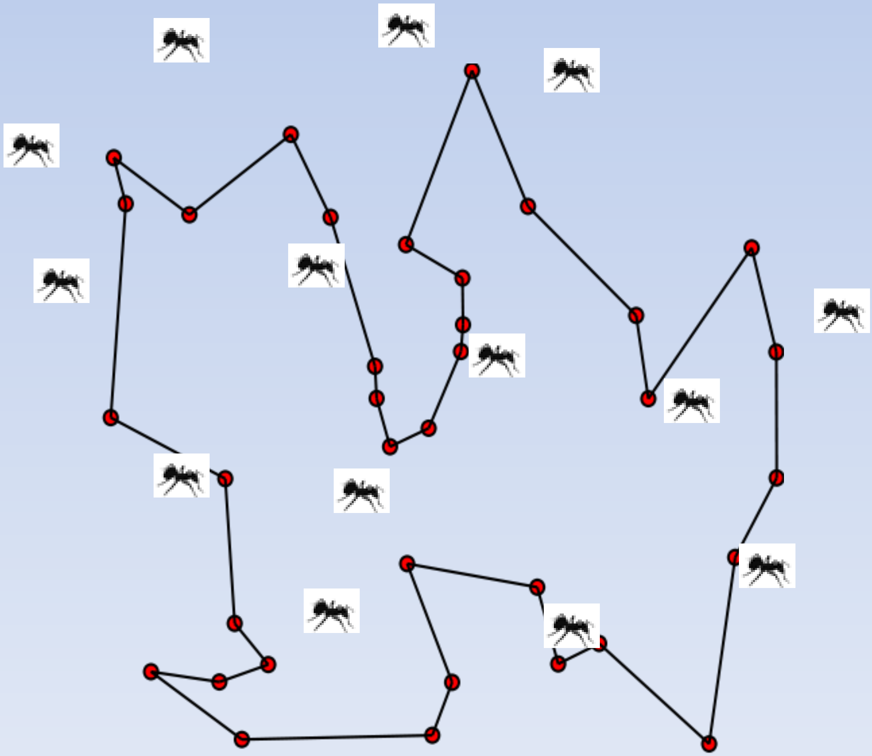
- **Travelling Salesman Problem** (**TSP**): *"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?*



- NP-hard problem in combinatorial optimization

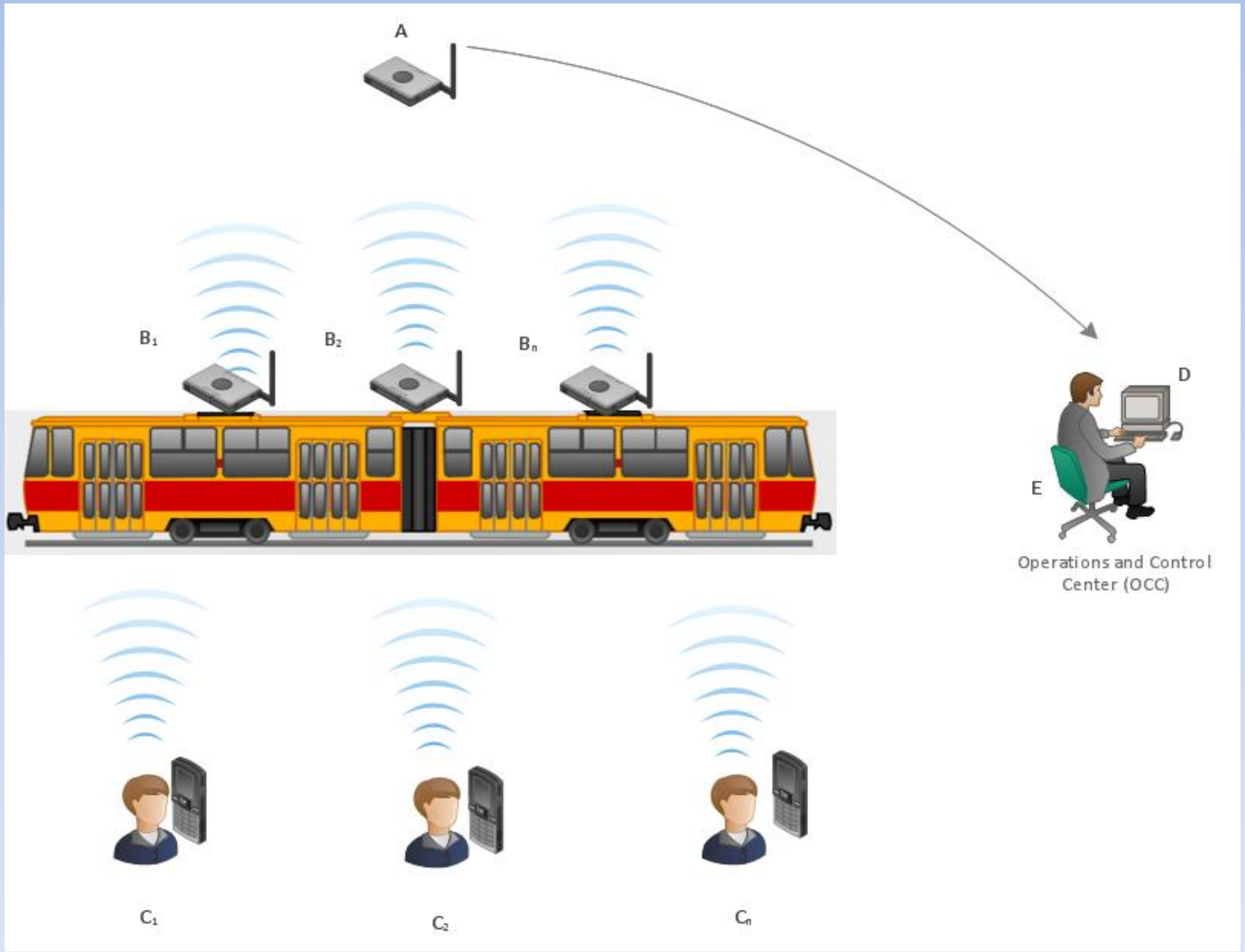
5. Feasible application for Smart City (III)

- Travelling Salesman Problem (TSP) + Ant Colony Optimization (ACO)
- **Massive computation ☹**
- **Ants can be used to trim the search tree down ☺**



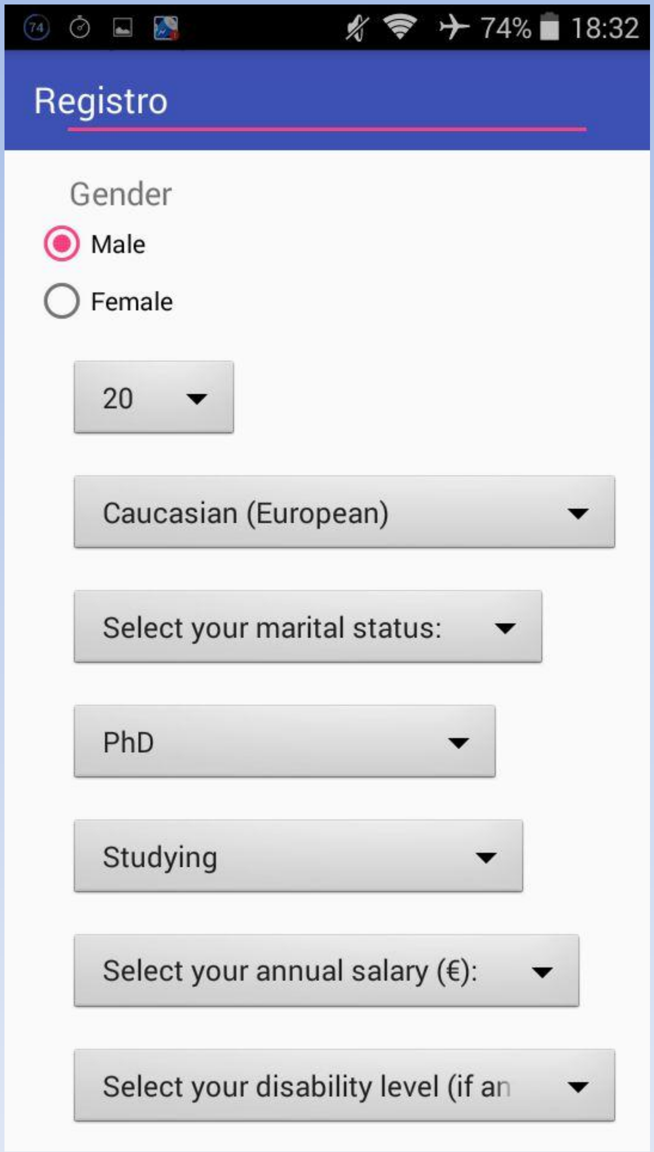
5. Feasible application for Smart City (IV)

- System for endowing intelligence to the UPTS in development:



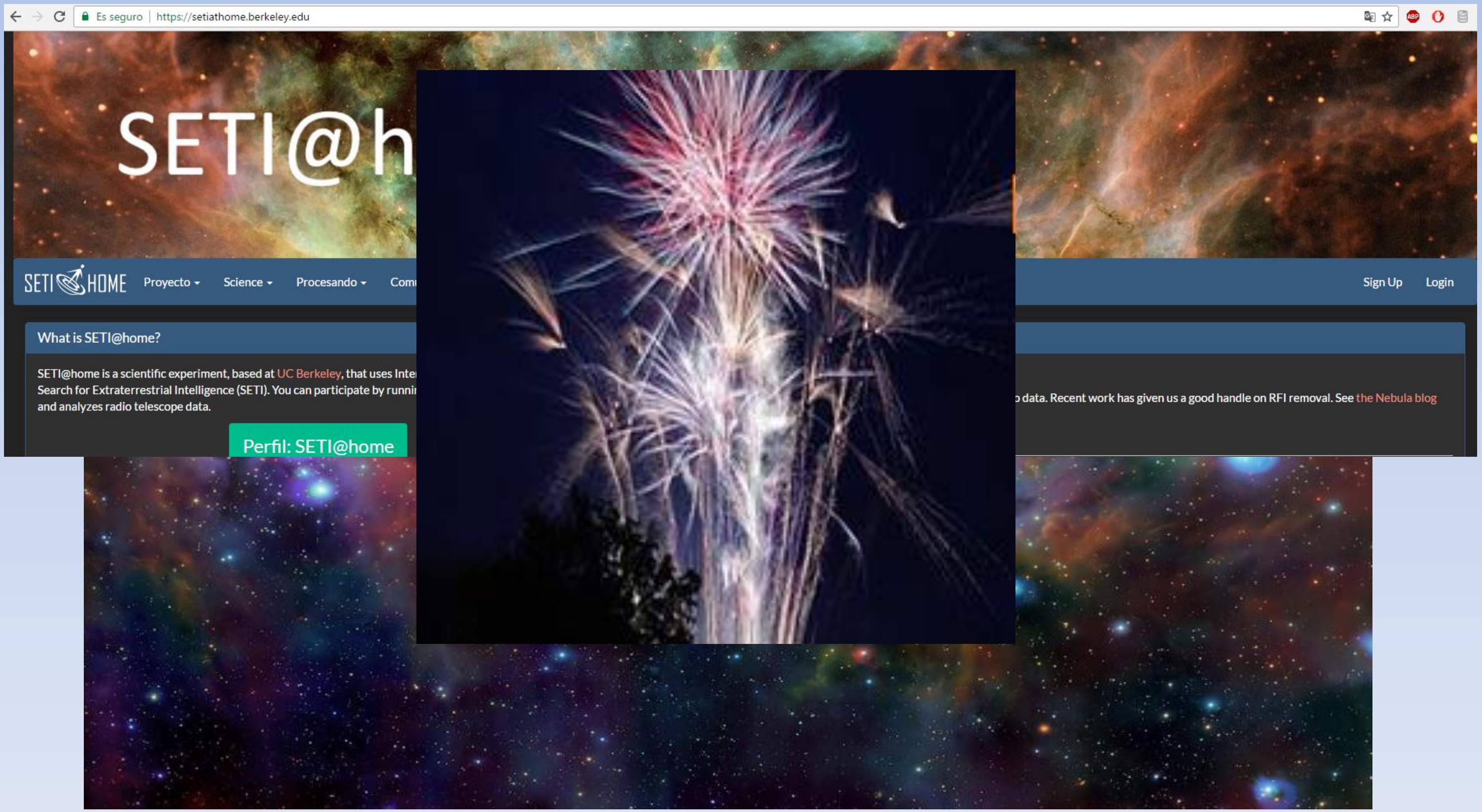
5. Feasible application for Smart City (V)

- **Genetic Algorithms** to **evolve** a route. Mobile application to:
  - **Inform users about backup routes** in case of systems breakdown
  - **Calculate route** between two points, even if there are blocked sections due to failure
  - User **super-pheromone** for statistical use



5. Feasible application for Smart Space?(VI)

- **Search for Extraterrestrial Intelligence (SETI)**



Index of Contents

1. The Natural Computing Paradigm & Smart Cities
2. Fireworks Algorithm
3. Current Research
4. Investigation Results
5. Fireworks Algorithm Applications
6. **Conclusions & Discussion**

6. Conclusions & Discussion

- **Citizens** can be seen as **particles** among a huge **swarm**. Natural Computing algorithms are highly mixable.

- PhD investigation efforts currently centered in the **Fireworks Algorithm and an optimization has been reached**
  - **FWA with pre-initialization is a 24% faster :-)**
  - **Investigating synergy between FWA and ACO**

6. Conclusions & Discussion

Base

- The Natural Computation paradigm has been investigated, as well as its application in Smart Cities spectrum

Investigation

- A FWA optimization based on pre-initialization scheme has been given, as well as a system for finding & rescuing lost people in rural areas (FWA+ACO)

Development

- FWA speed has been increased by 24% and the synergy between FWA+ACO looks promising.

Seminario de Investigación 17-18

Fireworks Algorithm applied to Smart Cities field & Optimizations

Universidad Politécnica de Madrid

ETSI SISTEMAS INFORMÁTICOS

PhD Candidate: **Clemencio Morales Lucas**
Contact: mail@clemenciomorales.com

PhD Director: **Dr. Luis Fernando de Mingo López**